Efficient characterisation of large deviations using population dynamics

Tobias Brewer[1], Stephen R. Clark[1,2] Russell Bradford[3], and Robert L. Jack[1,4,5],

[1] Department of Physics, University of Bath, Bath BA2 7AY, United Kingdom
[2] Max Planck Institute for the Structure and Dynamics of Matter, University of Hamburg CFEL, Hamburg, Germany
[3] Department of Computer Science, University of Bath, Bath BA2 7AY, United Kingdom
[4] Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Wilberforce Road, Cambridge CB3 0WA, United Kingdom
[5] Department of Chemistry, University of Cambridge, Lensfield Road, Cambridge CB2 1EW, United Kingdom

Abstract. We consider population dynamics as implemented by the cloning algorithm for analysis of large deviations of time-averaged quantities. We use the simple symmetric exclusion process with periodic boundary conditions as a prototypical example and investigate the convergence of the results with respect to the algorithmic parameters, focussing on the dynamical phase transition between homogeneous and inhomogeneous states, where convergence is relatively difficult to achieve. We discuss how the performance of the algorithm can be optimised, and how it can be efficiently exploited on parallel computing platforms.

1. Introduction

Large deviation theory [1] is concerned with the probabilities of rare events in random processes. Despite their scarcity, such events can have dramatic consequences, especially if one considers the behaviour of systems on long time scales, as in geology [2] or climate science [3]. In theoretical physics, a number of recent studies (for example [4, 5, 6, 7, 8]) have concentrated on rare events in which a system does not behave ergodically – that is, time-averaged observable quantities have non-typical values, even when measured over long time periods. While, it is challenging to capture rare events in experiments (see however [9]), there are a range of computational approaches for the characterisation of rare events [10, 11, 12, 13, 14, 15, 16, 17] – these are vital since analytical calculations are usually possible only in very simple systems [4]. Computational analyses of large deviations using these methods have lead to new insights in glassy materials [18, 19], protein-folding [8, 20, 21] and integrable systems [22]. In particular, analysis of the associated rare-event mechanisms can reveal properties of metastable or unusual dynamical states that can aid understanding of the typical behaviour.

When considering the large deviations of time-averaged quantities, the two dominant computational methods are transition path sampling [10] and a cloning (or population dynamics) algorithm [12]. This article investigates properties of the cloning method, applied to the symmetric simple exclusion process (SSEP). This is a prototypical interacting-particle system, for which analytical results are available [4, 23, 24], allowing a direct comparison between theory and simulation. While the model is very simple, it exhibits a range of surprising rare-event phenomena, including events where many particles assemble into a large (macroscopic) cluster, as well as hyperuniform states [25], where density fluctuations are strongly suppressed. These regimes of behaviour are separated by dynamical phase transitions [26, 27, 28]: at these points, numerical calculations become challenging and require analysis of large system sizes and long time scales.

The cloning algorithm was introduced more than ten years ago [12, 29], based on earlier ideas in statistical physics [30, 31] and quantum mechanics [32]. It has been applied to a range of systems [22, 33, 34, 35]. The method is powerful, but it requires simulations of many copies ("clones") of the system, and its results are accurate only in the limit where the number of clones

tends to infinity. For finite numbers of clones, there are both systematic and random errors, which have been analysed recently in Refs. [36, 37, 38]. The scaling of these errors has been determined using an analytical description of the algorithm and has been verified using a numerical approach to measure how quickly an estimator converges towards its true value [36]. Simple guiding models (or control forces) have been used to improve the efficiency of convergence [39, 35, 38]. In previous analyses of the convergence of the algorithm, it was often assumed that the clone population is larger than the total number of states visited by the model [36, 37]: this is not the case in typical applications so we analyse here the case where the clone population is much smaller than the total number of states.

We have implemented the cloning method on high-performance computing platform, which allows us to investigate large numbers of clones. We present a detailed analysis of the dynamical phase transition that occurs in the SSEP. The finite-size scaling properties of this system differ from conventional phase transitions [23, 24], and we discuss the physical reasons for this. In addition, we discuss the systematic errors inherent in the cloning method; we provide practical heuristics as to how the significance of these errors can be assessed; we provide some simple optimisations of the method in order to reduce these errors. Finally, we discuss our computational implementation, and how this can be optimised to improve performance when the number of clones is large.

The form of this paper is as follows: Section 2 provides background on large deviation theory and the cloning method; Section 3 describes the model and its dynamical phase transition, including some numerical results that allow characterisation of finite-size effects. Section 4 explains in more detail how we apply the method to the SSEP, and how this method is optimised to reduce errors. After that, Section 5 analyses the convergence of the algorithm with respect to the number of clones, and the (long) time scale associated with the rare events of interest. Finally, Section 6 discusses the computational implementation, and we summarise our conclusions in Section 7.

2. Large deviation theory and the cloning algorithm

2.1. Large deviation theory for time averaged-quantities

Consider a physical system described by a Markov process on a discrete set of states (for example, the SSEP). The state of the system at time t is $\mathcal{C}_t$, the transition rate from state $\mathcal{C}$ to $\mathcal{C}'$ is $W(\mathcal{C} \to \mathcal{C}')$, and we define the *escape rate* as $r(\mathcal{C}) = \sum_{\mathcal{C}'} W(\mathcal{C} \to \mathcal{C}')$. Let Θ denote a trajectory of the system, during the time interval $[0, t_{\rm obs}]$, for some *observation time* $t_{\rm obs}$. In trajectory Θ, suppose that the configuration changes happen at times $t_1, t_2, \ldots t_K$, and let the state of the system just after the kth change be $\mathcal{C}_k$ (also let the initial configuration be $\mathcal{C}_0$ and define $t_{K+1} = t_{\rm obs}$ and $t_0 = 0$). Then, denoting the probability of trajectory Θ by $P(\Theta)$, one has (see for example [19]):

$$P(\Theta) = \left[\prod_{k=0}^{K-1} W(\mathcal{C}_k \to \mathcal{C}_{k+1})\right] \cdot \exp\left[-\sum_{k=0}^{K} r(\mathcal{C}_k)(t_{k+1} - t_k)\right]. \tag{1}$$

Now let A_t be a (random) observable quantity that depends on the behaviour of the system during the time-interval $[0, t]$. For example, in the SSEP, A_t will be the total number of particle hopping events in $[0, t]$, as discussed in [4, 25]. Alternatively, A_t might be a time integral of the form $\int_0^t b(\mathcal{C}_{t'})\mathrm{d}t'$, where b is some function that depends on the configuration. With either of these choices, one expects that for large t, the probability distribution of A_t scales as

$$\mathrm{Prob}(A_t \approx at) \sim \exp(-\pi(a)t). \tag{2}$$

This is an example of a large-deviation principle [1, 19]. The function π is known as the rate function, and satisfies $\pi(a) \geq 0$. Typically, there is a single value $\overline{a}$ for which $\pi(\overline{a}) = 0$. In that case, one sees from (2) that as $t \to \infty$, the distribution of $a_t = A_t/t$ concentrates on the single value $\overline{a}$, with the probability of any other value being suppressed exponentially in t. Of course, the validity of (2) depends on the system of interest and the observable A – here we consider irreducible Markov processes with finite (discrete) state spaces, for which (2) holds for a large set of observables A_t: see eg [19].

The general aims of rare-event sampling methods in this context are (i) to estimate the function $\pi(a)$, which gives the probability of rare events (with $a_t \neq \overline{a}$); and (ii) to characterise the

rare events themselves: what trajectories lead to these rare values? To achieve these aims, it is convenient to introduce a *biasing field* – denoted by s – which allows access to the relevant rare events. We use Θ to denote a trajectory of the system, during the time interval $[0,\ t]$, and let $P(\Theta)$ be the probability of trajectory Θ as defined, for example in [19]. The distribution $P(\Theta)$ depends on the initial condition of the model. The results of the following large-deviation analysis are independent of the initial condition, but we assume for concreteness that the initial condition is taken from the steady-state probability distribution of the model, so that $P(\Theta)$ corresponds to the steady state.

Now define a new probability distribution

$$\tilde{P}_t(\Theta, s) = \frac{P(\Theta)\exp[-sA_t(\Theta)]}{Z(s,t)}, \tag{3}$$

where $A_t(\Theta)$ is the value of A_t associated with trajectory Θ, and $Z(s,t) = \left\langle e^{-sA_t}\right\rangle_0$ is a dynamical partition function (normalisation constant). Here and throughout, the notation $\langle\cdot\rangle_0$ indicates an average with respect to $P(\Theta)$. The average of an observable with respect to $\tilde{P}_t$ is denoted by

$$\langle\mathcal{O}\rangle_s = \int \mathcal{O}(\Theta)\tilde{P}_t(\Theta, s)d\Theta. \tag{4}$$

It is useful to define

$$\psi(s) = \lim_{t\to\infty}\frac{1}{t}\ln Z(s,t). \tag{5}$$

This limit certainly exists if the LDP (2) holds.

The distribution $\tilde{P}$ is parameterised by the field s. For $s = 0$ we recover the original distribution P; for $s > 0$ trajectories with large values of A_t are suppressed. The advantage of introducing the field s is that averages of the form $\langle\mathcal{O}\rangle_s$ can often be evaluated by some numerical or analytical method. In the absence of dynamical phase transitions, one may then obtain the rate function in (2), as $\pi(a) = \max_s[-sa - \psi(s)]$. Moreover, if the value of s that achieves this maximum is s_a^* then trajectories obtained from the distribution $\tilde{P}(\Theta, s_a^*)$ are representative trajectories associated with the rare event $a_t = a$ discussed above in the large time limit [40]. Thus, computational analysis of $\tilde{P}$ can achieve the two aims (i) and (ii) above, to estimate π and to characterise the trajectories that realise these rare events.

The function ψ is a scaled cumulant generating function [19]: one has

$$\lim_{t\to\infty}\langle A_t/t\rangle_s = -\psi'(s) \tag{6}$$

where the prime denotes a derivative. There is also an associated susceptibility (scaled variance)

$$\lim_{t\to\infty}\frac{1}{t}\left[\left\langle A_t^2\right\rangle_s - \langle A_t\rangle_s^2\right] = \psi''(s). \tag{7}$$

2.2. Modified Dynamics

Large deviations are hard to analyse computationally because the associated events are rare – this means that averages such as $\langle \mathrm{e}^{-sA_t}\rangle_0$ are dominated by trajectories that are not at all typical of the system at equilibrium. To analyse such events, it is often convenient to modify the dynamics of the model, so that the relevant trajectories become less rare [12, 39, 35]. Consider a general system with modified (or "controlled") dynamics, and let its path probability distribution be $\hat{P}(\Theta)$, which is analogous to the distribution $P(\Theta)$ for the original system. For the modifications that we consider, it is possible to relate these two distributions, as

$$P(\Theta) = \hat{P}(\Theta)\mathrm{e}^{-\hat{U}_t(\Theta)} \tag{8}$$

where $\hat{U}_t(\Theta)$ is a weight function that depends on the trajectory Θ. (A specific example will be considered in Sec. 3.2, below.) Hence one has also

$$\tilde{P}_t(\Theta, s) = \frac{1}{Z(s,t)}\hat{P}(\Theta)\mathrm{e}^{-[\hat{U}_t(\Theta)+sA_t(\Theta)]}. \tag{9}$$

The significance of this result is that the distribution $\tilde{P}_t$ can be analysed in many different ways: either directly as in (3) or by simultaneously modifying the dynamics of the system, $P(\Theta) \to \hat{P}(\Theta)$, and at the same time modifying the weighting factor as $sA_t(\Theta) \to [sA_t(\Theta) + \hat{U}_t(\Theta)]$. This freedom to modify the dynamics is very useful when designing computational algorithms. Finally, we define

$$\Upsilon_t(\Theta) = \exp[-\hat{U}_t(\Theta) - sA_t(\Theta)], \tag{10}$$

which is the weight that should be associated with trajectory Θ, in order to obtain the distribution (3) by importance sampling from the distribution $\hat{P}$. In particular, we have

$$Z(s,t) = \int \Upsilon_t(\Theta)\hat{P}(\Theta)d\Theta. \tag{11}$$

2.3. Cloning algorithm

The results of this paper use the cloning (or population dynamics) algorithm that was proposed by Giardina, Kurchan and Peliti [12] as a method for studying large deviations. As noted above, this method draws on earlier work by Grassberger [31] as well as Diffusion Quantum Monte Carlo methods [32, 38]. We outline this algorithm here, further details are provided in Section 4, below. The method is based on simulations of a population of $n_{\rm c}$ copies (or clones) of the system, evolving over a total observation time $t_{\rm obs}$. The field s is a fixed parameter: to obtain accurate estimates of $\psi(s)$ one requires a limit of large $n_{\rm c}$ and $t_{\rm obs}$. The dependence of the results of the algorithm on $n_{\rm c}$ and $t_{\rm obs}$ will be discussed in Section 5 below. In our implementation, the population size is held strictly constant, although modified algorithms with variable populations are also possible [29].

Within the algorithm, the total time $t_{\rm obs}$ is split into intervals of length Δt, so the number of such intervals is $M = t_{\rm obs}/\Delta t$. Within each step of the algorithm, each clone evolves independently for a time Δt. Then, some clones are deleted and others copied, in order to bias the system towards the rare events of interest (this is a form of importance sampling). In the following we refer to these two sub-steps (or stages) as the dynamical stage and the cloning stage of the algorithm. The full algorithmic step – dynamics followed by cloning – is repeated M times. The parameter Δt can be chosen according to the problem of interest: as $n_{\rm c} \to \infty$ (with fixed $t_{\rm obs}$) then the results are independent of Δt. However, Δt has significant effects on the accuracy of the results obtained: this is discussed in Section 4.3 below.

We index the time intervals by $\beta = 1, 2, \ldots M$ and define $t_\beta = \beta\Delta t$. Then the cloning method rests on the fact that for any trajectory Θ, one may write

$$A_{t_{\rm obs}}(\Theta) = \sum_{\beta=1}^{M} A^\beta(\Theta)$$

where $A^\beta(\Theta)$ is the contribution to $A_t(\Theta)$ from the time interval $[t_{\beta-1}, t_\beta]$. Similarly

$$\hat{U}_{t_{\rm obs}}(\Theta) = \sum_{\beta=1}^{M} \hat{U}^\beta(\Theta). \tag{12}$$

Note that in the definitions of $A^\beta(\Theta)$, $\hat{U}^\beta(\Theta)$ the superscript β is an index and should not be confused with an exponent. We also index the clones of the system by $i = 1, 2, \ldots n_{\rm c}$. Then, define a weighting factor for clone i associated with time-interval β as

$$\Upsilon^\beta(\Theta_i) = \exp[-\hat{U}^\beta(\Theta_i) - sA^\beta(\Theta_i)]. \tag{13}$$

where Θ_i is the trajectory followed by clone i. This weighting factor plays two roles within the algorithm. First, in the importance sampling step that takes place at time t_β, the average number of times that clone i is copied is proportional to $\Upsilon^\beta(\Theta_i)$. Second, based on (5,11), one may estimate $\psi(s)$ as

$$\hat{\psi}(s) = \frac{1}{t_{\rm obs}} \sum_{\beta=1}^{M} \ln\left(\frac{1}{n_{\rm c}} \sum_{i=1}^{n_{\rm c}} \Upsilon^\beta(\Theta_i) \right). \tag{14}$$

For a given computation, this estimator is subject to both systematic and random errors. However, both these errors vanish as $n_{\rm c}, t_{\rm obs} \to \infty$, and the estimator becomes exact.

2.4. Estimating averages with respect to $\tilde{P}$.

To estimate averages of the form $\langle \mathcal{O} \rangle_s$, one starts by considering the population of clones at the final time $t_{\rm obs}$. For each clone i in that population, one follows its trajectory backwards in time: this trajectory is denoted by $\hat{\Theta}_i$. Note that many members of the final clone population may be descended from a single clone at some earlier time. Hence, the trajectories $\hat{\Theta}_i$ are not all independent samples from $\tilde{P}$. However, one may estimate the general expectation value (4) as

$$\langle \mathcal{O} \rangle_s \approx \frac{1}{n_{\rm c}} \sum_{i=1}^{n_{\rm c}} \mathcal{O}(\hat{\Theta}_i) \tag{15}$$

where $\mathcal{O}(\Theta)$ is the value of observable $\mathcal{O}$ in trajectory Θ. The approximate equality becomes exact [36] as $n_{\rm c} \to \infty$.

The observable $\mathcal{O}$ may depend in general on all aspects of the trajectory Θ. It is also useful to consider a specific class of time-dependent observables: let F_t be a function that depends on the state of the system at time t, such as the number of particles on a particular lattice site. The average of such an observable is $\langle F_t \rangle_s$, which may be evaluated for any time t between 0 and $t_{\rm obs}$. For $s = 0$, the probability distribution P is time-translation invariant (TTI), which means that $\langle F_t \rangle_0$ does not depend on t. However, for $s \neq 0$, the average $\langle F_t \rangle_s$ depends on t: there are initial and final transient regimes for small t and for $t \approx t_{\rm obs}$, with an intermediate time-translation invariant regime. That is,

$$\langle F_t \rangle_s = \begin{cases} F_{\rm i}(t), & t \lesssim \tau \\ F_{\rm f}(t_{\rm obs} - t), & (t_{\rm obs} - t) \lesssim \tau \\ F_\infty, & \text{otherwise} \end{cases} \tag{16}$$

where τ is a characteristic time scale for the transient regimes, and $F_{\rm i}, F_{\rm f}$ are functions describing the transients, which decay to the asymptotic value F_∞ as their arguments get large [41, 39].

It will be useful in the following to consider the probability distribution of F_t, and not just its mean value. This distribution is defined as

$$p_{t_{\rm obs}}(F, t) = \langle \delta(F - F_t) \rangle_s,$$

which is the probability (density) to observe the value F for the observable F_t, given trajectories of length $t_{\rm obs}$ with distribution $\tilde{P}$. To characterise the TTI regime, we define

$$p_{\rm ave}(F) = \lim_{t_{\rm obs} \to \infty} p_{t_{\rm obs}}(F, \alpha t_{\rm obs}) \tag{17}$$

with $0 < \alpha < 1$. The result is independent of α because we have both $\alpha t_{\rm obs} \to \infty$ and $(t_{\rm obs} - \alpha t_{\rm obs}) \to \infty$. Hence, αt is a time in the TTI regime, αt, $(t_{\rm obs} - \alpha t) \gg \tau$ and it follows that $F_\infty = \int F p_{\rm ave}(F) {\rm d}F$. We also define

$$p_{\rm end}(F) = \lim_{t_{\rm obs} \to \infty} p_{t_{\rm obs}}(F, t_{\rm obs}) \tag{18}$$

which is the distribution of F at time $t = t_{\rm obs}$.

Note that we have focussed here on instantaneous observables: F_t depends only on the configuration of the system at time t. However, the definitions of $p_{\rm ave}, p_{\rm end}$ can be straightforwardly generalised to observables that depend on the trajectory of the system, within a finite time window $[t, t + \Delta t]$.

Within the cloning algorithm, the relevance of $p_{\rm ave}$ and $p_{\rm end}$ is that the clone population just after the importance sampling step is distributed as $p_{\rm end}$. On the other hand, the distribution $p_{\rm ave}$ characterises the "ancestral population": this is the distribution that one obtains by constructing the trajectories $\hat{\Theta}$ from the current population by following their histories backwards in time. For a detailed discussion see Ref [39].

3. Dynamical phase transition in the symmetric simple exclusion process

In the following, we apply the cloning algorithm to a model system: the symmetric simple exclusion process (SSEP).

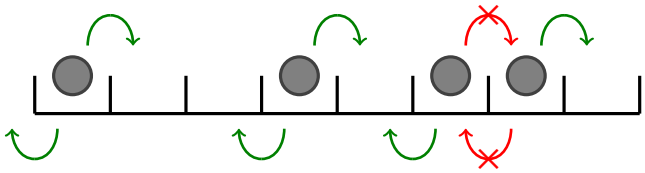

Figure 1: Illustration of the SSEP on a one-dimensional lattice of 8 sites with periodic boundary conditions and $N = 4$ particles. Each hop is attempted with rate 1; all possibilities for the attempted hop are indicated with arrows.

3.1. Model and choice of dynamical observable

We consider a one-dimensional lattice of L sites with periodic boundaries. The lattice is occupied by N particles with at most one particle per site. Each particle attempts to hop with rate 1 to each of its neighbouring sites as in Figure 1. The attempted hop is successful if the neighbouring site is unoccupied. Let the occupancy of site i be n_i. The model obeys detailed balance which means that in its equilibrium (steady) state, the occupancy of each site is independent: $n_i = 1$ with probability $\rho = N/L$ and $n_i = 0$ with probability $1 - \rho$.

For any trajectory Θ, define the activity $K_t(\Theta)$ as the total number of (successful) hops in the time-interval $[0, t]$. Within the steady state of the model, one has $\langle K_t/t\rangle_0 = 2L\rho(1-\rho)$, since the rate of attempted hops is $2N = 2\rho L$ and the expected fraction of successful hops is equal to the probability $(1-\rho)$ that the destination site is unoccupied. We consider the distribution $\tilde{P}$ defined as in (3), with $A_t = K_t$: from (1) one has

$$\tilde{P}_{t_{\rm obs}}(\Theta, s) = \frac{1}{Z(s, t_{\rm obs})} \left[\prod_{k=0}^{K-1} W(\mathcal{C}_k \to \mathcal{C}_{k+1}) \mathrm{e}^{-s}\right] \cdot \exp\left[-\sum_{k=0}^{K} r(\mathcal{C}_k)(t_{k+1} - t_k)\right].$$

3.2. Modified dynamics

To improve computational efficiency when sampling from $\tilde{P}$, it is useful to adopt a simple modification to the dynamics, as described in Section 2.2. In this modification, all transition rates are rescaled by a factor e^{-s}.

From (1), the resulting probability distribution is

$$\hat{P}(\Theta) = \left[\prod_{k=0}^{K-1} W(\mathcal{C}_k \to \mathcal{C}_{k+1}) \mathrm{e}^{-s}\right] \cdot \exp\left[-\sum_{k=0}^{K} r(\mathcal{C}_k)\mathrm{e}^{-s}(t_{k+1} - t_k)\right].$$

Hence, from (10),

$$\Upsilon_{t_{\rm obs}}(\Theta) = \exp\left[-\sum_{k=0}^{K} r(\mathcal{C}_k)(1 - \mathrm{e}^{-s})(t_{k+1} - t_k)\right]. \tag{19}$$

In the following, we use these modified dynamics and the weight factors Υ within our implementation of the cloning algorithm. This modification to the dynamics is useful because the biasing field s has the effect of suppressing all transitions, so as to reduce K. Since the modified dynamics incorporates this (trivial) effect, the resulting trajectories are closer to the biased trajectories of interest than one would get by simulating the SSEP directly.

3.3. Dynamical phase transition

Our motivation for studying large deviations of K_t in the SSEP is twofold. First, the model is simple enough for a precise numerical characterisation, and is a useful test of the numerical method. Second, there is a dynamical phase transition that takes place in the model [27, 26, 24]

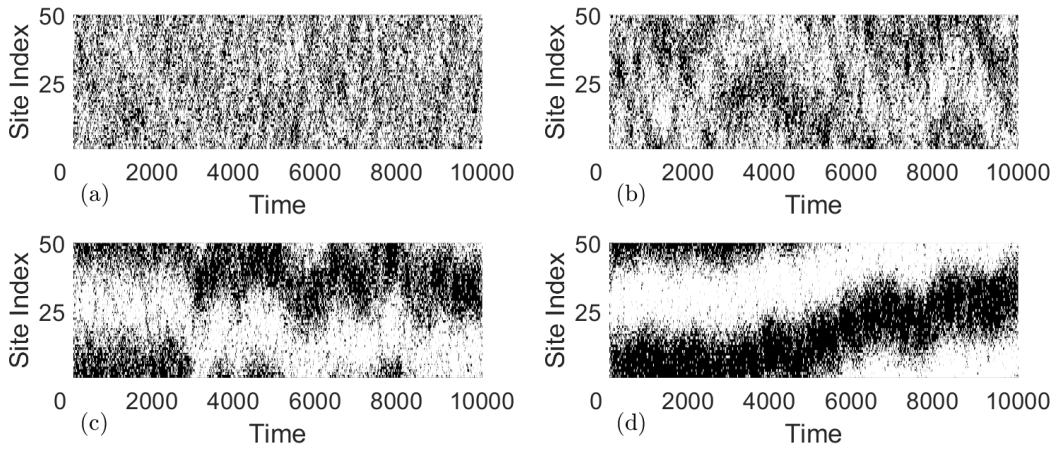

Figure 2: Sample trajectories of SSEP with $L = 50$, $N = L/2$ and $t_{\rm obs} = 10^4$. (a) $s = 0$, the equilibrium state; (b) $s = 0.008$, showing evidence of transient clusters; (c) $s = 0.012$, in which a single large cluster has formed; (d) $s = 0.020$, with most of the particles in a well-defined single cluster. The corresponding values of λ are $0, 20, 30, 50$ and the critical value of λ is $\lambda_c = 2\pi^2 \approx 19.7$.

which reveals interesting physical effects. To investigate the phase transition, it is useful to scale the bias s by the square of the system size: we define

$$\lambda = sL^2 \tag{20}$$

The phase transition takes place at a critical value of λ, and leads to numerical challenges that we use in later sections to test the cloning method.

The phase transition appears if one fixes the density $\rho = N/L$ and takes the lattice size $L \to \infty$. [We take this limit after the large-t limit associated with the large deviation principle (2).] For finite L, the rate function $\pi(a)$ and the cumulant generating function $\psi(s)$ in (5) are both analytic functions, and there is no phase transition. However, on taking $L \to \infty$, a singular response to the field s can be observed, just as conventional phase transitions can be observed on taking the thermodynamic limit. Specifically, one considers $\psi_*(s) = \lim_{L\to\infty}[\psi(s)/L]$, which is analogous to a thermodynamic free energy density, whose derivatives show singular behaviour at phase transitions.

The physical signature of this transition is shown in Figure 2 where there is a transition from a homogeneous state at $s = 0$ (particles are distributed evenly throughout the system) to an inhomogeneous ("phase-separated") state for $s > 0$, in which case the particles are segregated into a dense and a dilute region.

To quantify the particle clustering that takes place for $s > 0$, it is useful to consider the Fourier transform of the density field:

$$\delta\rho_n = \frac{1}{\sqrt{L}} \sum_{j=1}^{N} e^{-2\pi i n X_j/L} \tag{21}$$

where X_j is the index of the site occupied by particle j, and $n = 0, 1, \ldots L-1$. We focus on the wave vector that corresponds to the longest wavelength fluctuations, that is $n = 1$.

Figure 3 shows that the magnitude of this Fourier component grows as the system becomes inhomogeneous.

3.4. Scaling at the dynamical phase transition

To investigate this phase transition in more detail, it is useful to "zoom in" on the crossover from the homogeneous to the inhomogeneous case. To achieve this, let $s = \lambda/L^2$ as in (20) and consider

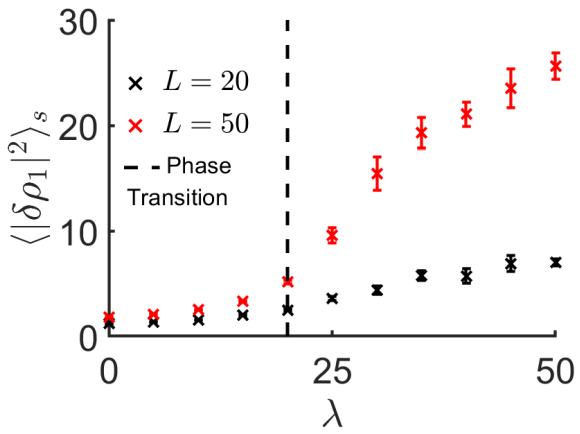

Figure 3: The mean square value of the first Fourier component of the density, $\langle|\delta\rho_1|^2\rangle_s$, measured at $t = t_{\rm obs}/2$ with $t_{\rm obs} = 10^4, n_{\rm c} = 10^5$, for various L, λ. The phase transition occurs at $\lambda_c = 2\pi^2$: for $\lambda > \lambda_c$ one expects the system to become inhomogeneous, so that $\langle|\delta\rho_1|^2\rangle_s \propto L$, consistent with the data.

the limit of large L at fixed λ. This is analogous to finite-size scaling in equilibrium systems. We focus in this work on the crossover function

$$\mathcal{K}_L(\lambda) = k(\lambda/L^2), \qquad \text{where} \qquad k(s) = L^{-1} \lim_{t\to\infty} \langle K_t/t\rangle_s = -\psi'(s)/L. \tag{22}$$

As $L \to \infty$, the function $\mathcal{K}_L$ converges to a limiting form $\mathcal{K}_*$, which can be computed using macroscopic fluctuation theory [24]. This function has a singularity at $\lambda = \lambda_c = 2\pi^2$. For $\lambda < \lambda_c$, one has a constant value $\mathcal{K}_* = 2\rho(1-\rho)$. For $\lambda > \lambda_c$, the function $\mathcal{K}_*$ decreases with λ, converging to zero as $\lambda \to \infty$. This behaviour is illustrated in Figure 4, which also shows the rescaled "free energy"

$$\phi_L(\lambda) = L\psi(\lambda/L^2)$$

To show the singularity that appears at the phase transition, we also show two measures of susceptibility

$$\chi_L(s) = L^{-1}\psi''(s), \qquad \mathcal{X}_L(\lambda) = -\mathcal{K}_L'(\lambda) = L^{-2}\chi_L(\lambda/L^2). \tag{23}$$

The bare susceptibility χ_L corresponds to the scaled variance of K_t (recall (7)), and $\lim_{L\to\infty}\chi_L(s)$ should have a finite value in a system that is away from any phase transition. Figure 4 shows a divergence in χ, consistent with the existence of a phase transition. On the other hand, the function $\mathcal{X}_L(\lambda)$ is predicted by MFT to have a finite limit as $L \to \infty$. Our numerics are consistent with this prediction but they show that measuring this limiting function requires large system sizes.

Note that the scenario illustrated in Figure 4 is different from classical finite-size scaling and from other first-order dynamical phase transitions [35]. Comparing with the classical case, note that we take $t_{\rm obs} \to \infty$ and then later $L \to \infty$. As discussed in Ref. [35], this is equivalent to thermodynamic finite-size scaling in a cylindrical geometry, with the length of the cylinder being much longer than its perimeter. In that case, one possibility is that the susceptibility χ_L grows exponentially in the system size, due to the a distribution of domains along the cylinder, with "domain walls" perpendicular to the long axis of the cylinder. However, the results presented are very different from that case: one reason is that the coexisting phases at the transition have different densities, but the number of particles in the SSEP is equal at every time. As a result, the "domain walls" between dense and dilute regions in Figure 2 are constrained to lie parallel to the time axis. In general, χ_L can be interpreted as a "correlation volume" in space-time. From (23), and noting that $\mathcal{X}(\lambda)$ is finite (or zero) for all λ [24], one sees that χ_L diverges as L^2 in the vicinity of the transition. This factor arises from the characteristic time scale proportional to L^2 that is associated with density fluctuations on length scale L. Note also that while $k(s)$ exhibits a jump at $s = 0$, corresponding to a first-order phase transition, one may also consider the

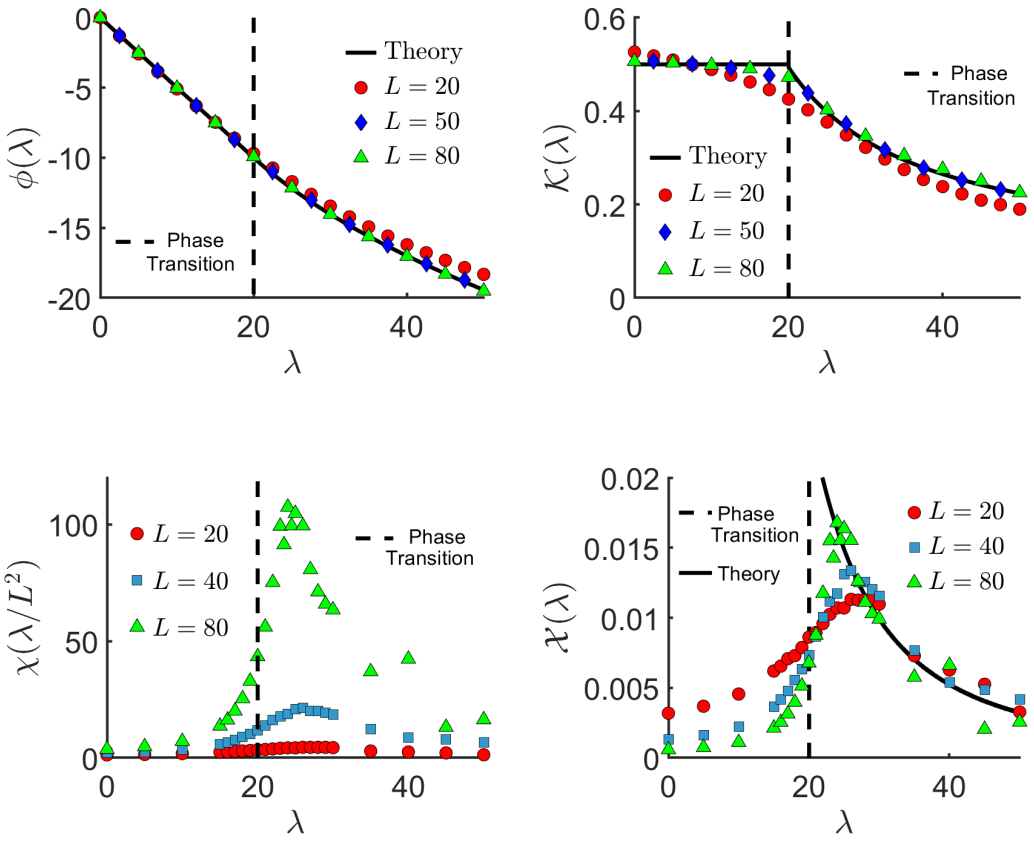

Figure 4: Results illustrating the dynamical phase transition in the SSEP. The number of clones is $n_c = 10^5$ for $L = 20, 40, 50$ and $n_c = 10^6$ for $L = 80$. The cloning interval is $\Delta t = 10$ and all results are averaged across 10 independent computations. The vertical dashed line shows the position of the phase transition ($\lambda_c = 2\pi^2$). The theoretical predictions are obtained by using finite differences to take derivatives of $\phi(\lambda)$, which is calculated as described in [24]. For $\lambda < \lambda_c$ one has $\mathcal{X} \to 0$ as $L \to \infty$: the leading behaviour for large L is $\mathcal{X} = O(1/L)$, and was computed in [23]. Our numerical results are consistent with that prediction (data not shown).

behaviour of the system as a function of λ, in which case $\mathcal{K}(\lambda)$ is continuous at the transition but has a discontinuous first derivative: when viewed on this scale, the transition has some features of a continuous phase transition [42, 43].

4. Implementation of algorithm and cloning stage

We outlined the cloning algorithm in Section 2.3. Here we provide some extra detail on its application to the SSEP. The modified SSEP dynamics are implemented using the Bortz-Kalos-Lebowitz (continuous time Monte Carlo) algorithm [44]. All possible particle hops have the same rate e^{-s}. These dynamics take place over the time interval $[t_{\beta-1}, t_\beta]$. During this time period, the factor $\Upsilon^\beta(\Theta_i)$ is calculated: based on (19,13) one may write

$$\Upsilon^\beta(\Theta_i) = \exp\left(-\int_{t_{\beta-1}}^{t_\beta} (1-\mathrm{e}^{-s}) r(\mathcal{C}_t^i) dt\right)$$

where $\mathcal{C}_t^i$ is the configuration of clone i at time t and $r(\mathcal{C})$ is the escape rate for configuration $\mathcal{C}$ under the original (unmodified) dynamics. The factors Υ appear in the estimate (14) for $\psi(s)$.

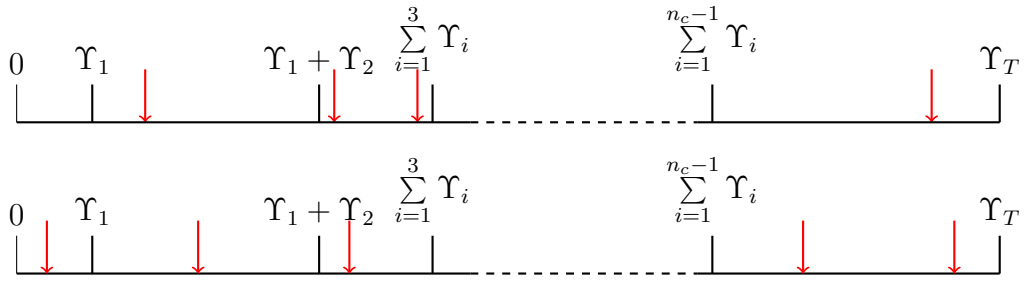

Figure 5: Example of the number line associated with the cloning factors $\Upsilon^\beta(\Theta_i)$ used in the clone selection methods, as described in the main text. We take $n_c = 1000$. The red arrows indicate some of the points α_j that determine which clone is selected for inclusion in the new population. The α may be chosen independently ([iid] method, top line) or equally spaced ([eq] method, lower line).

4.1. Copying and deletion of clones: [eq] and [iid] methods.

The next step is to clone and delete systems, according to their values of Υ, so as to produce a new population. There is some freedom as to how this is implemented within the algorithm. What we require is that for a large population the average number of descendants of clone i approaches $\Upsilon^\beta(\Theta_i)/\Upsilon_{\rm T}^\beta$ with $\Upsilon_{\rm T} = \sum_{i=1}^{n_c} \Upsilon^\beta(\Theta_i)$. We consider two methods for selecting the clones that will form the new population, both of which are consistent with this requirement.

The first clone selection method is denoted by [eq]: the reason for this will be explained below. In this method, for each clone j of the new population (with $1 \le j \le n_c$), we define $\alpha_j = (j + d - 1)\Upsilon_{\rm T}/n_c$, where d is a random number in $(0, 1)$ which is equal for each clone. Thus, the α_j are equally spaced on $[0, \Upsilon_{\rm T}]$, with $\alpha_1 = (d\Upsilon_{\rm T}/n_c)$ and $\alpha_{n_c} = \Upsilon_{\rm T} - [(1 - d)\Upsilon_{\rm T}/n_c]$. Then the state of clone j of the new population is pulled from clone k of the old population, where k satisfies

$$\sum_{i=1}^{k} \Upsilon^\beta(\Theta_i) \le \alpha_j < \sum_{i=1}^{k+1} \Upsilon^\beta(\Theta_i). \tag{24}$$

This clone selection method is equivalent to constructing a line segment of total length $\Upsilon_{\rm T}$ that is composed of contributions from each clone of the old population, with clone k contributing a length $\Upsilon^\beta(\Theta_k)$. Then clone j of the new population is selected by selecting the interval of the original line that contains the point α_j as in Figure 5. On average, the number of copies of clone k in the new population is then $\Upsilon^\beta(\Theta_k)/\Upsilon_{\rm T}$, as required. The label [eq] is used because the α_j are equally spaced on the interval $[0, \Upsilon_{\rm T}^\beta]$.

The second clone selection method that we use is denoted by [iid]. In this case, the α_j are identically and independently distributed (uniformly) on $[0, \Upsilon_{\rm T}^\beta]$, hence the label [iid]. However, this choice is less efficient, as we discuss below (Section 4.3). As noted in Sec. 2.3, the two sub-steps, of independent dynamical evolution followed by cloning, are each repeated M times, so that the total simulation time for each clone is $t_{\rm obs}$. The final step is a cloning step.

4.2. Test case

As a stringent test of algorithmic performance, we focus on the observable $\mathcal{K}_L(\lambda)$ defined in (22). We consider values of λ in the vicinity of the phase transition.

One sees from Figure 4 that this function has a feature at $\lambda = 2\pi^2$ that depends strongly on system size. In the following, we will test the ability of the algorithm to capture the finite-size scaling of this feature. Our numerical estimator for $\mathcal{K}_L$ is

$$\hat{k}_L(\lambda) = \frac{1}{Lt_{\rm obs}} \sum_{i=1}^{n_c} \sum_{\beta=1}^{M} K^\beta(\hat{\Theta}_i). \tag{25}$$

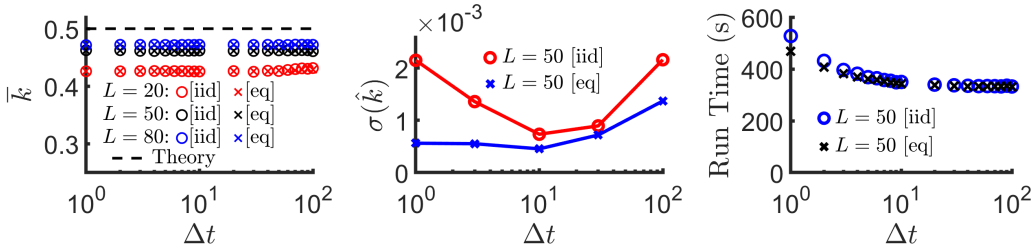

Figure 6: Dependence of results on the cloning interval τ and the clone selection mechanism ([eq] or [iid]) at the representative state point $\lambda = 28$ (see main text). (a) Analysis of the two clone selection mechanisms, showing that if the simulation is converged, the estimator $\overline{k}$ is (almost) independent of Δt, and of the clone selection method (as required). (b) The statistical errors are heavily dependent on the clone selection method and vary with Δt. (c) The computational time is almost independent of the clone selection method. It decreases with Δt because the amount of communication between nodes decreases, but this has a cost in terms of accuracy (see panel (b)).

see also (14): we again emphasise that in writing K^β then β is an index (not an exponent). Note, that computational evaluation of $\hat{k}$ requires that the trajectories $\hat{\Theta}$ can be obtained by extrapolation backwards in time: this is easily achieved by including with its clone its accumulated value of $\sum_\beta K^\beta(\hat{\Theta}_i)$, which is copied along with the system's configuration when the clone state is copied, during the cloning stage of the algorithm (see [45]). We use the method to measure $\mathcal{K}$ since this is a more revealing measure of algorithmic errors than $\psi(s)$ or $\phi(\lambda)$, as may be seen already in Figure 4. An alternative estimator of $\mathcal{K}$ can be obtained by using finite differences to estimate the derivative $\phi'(\lambda)$: we prefer the direct measurement (25), which avoids uncertainties arising from the finite differencing, although it does require an accurate sampling of the distribution $p_{\rm ave}(K^\beta)$ which may be challenging in practice: see Sec. 5.2 below.

In the following, we particularly focus on systematic errors: we average $\hat{k}$ over R independent computations (indexed by $r = 1, 2, \ldots R$) and we denote the average of the estimator by

$$\overline{k}_L(\lambda) = \frac{1}{R}\sum_{r=1}^{R} \hat{k}_L^r(\lambda),$$

and its variance by

$$[\Delta k_L(\lambda)^2] = \frac{1}{R}\sum_{r=1}^{R}\left[\hat{k}_L^r(\lambda)^2 - \overline{k}_L(\lambda)\right]^2.$$

Consequently, its standard deviation is $\sigma(\hat{k}) = [\Delta k_L(\lambda)^2]^{1/2}$. For large R, the systematic error of the method can be obtained as the difference between $\overline{k}_L(\lambda)$ and $\mathcal{K}_L(\lambda)$, and the size of the random error is determined by the variance $[\Delta k_L(\lambda)^2]$.

We emphasise that $\hat{k}_L(\lambda)$ is a simple estimate of $\mathcal{K}_L(\lambda)$ – other estimates might be obtained by running the algorithm with different parameters (for example a range of $n_{\rm c}$ or a range of λ), and combining the data in order to extrapolate or interpolate an improved estimate. See also Sec 5 below. However, when analysing the errors of the algorithm we concentrate on the direct estimate $\hat{k}_L$, since this can be analysed in a simple and precise way (for example, the use of a direct estimate means that there are no correlations of the errors between different state points in the following Figures).

4.3. Effect of cloning method and choice of Δt

Given a large enough number of clones, the results of the algorithm are independent of the parameter Δt, and they are also independent of whether the α_j are chosen to be equally spaced [eq] or uniformly at random [iid]. In this section, we show the effects that these choices have on the results obtained with finite $n_{\rm c}$. As a representative state point (unless otherwise stated) we

focus on $\lambda = 28$ and $t_{\rm obs} = 10^4$ with $\Delta t = 10$, but the results are qualitatively similar for other parameters too. (Note that in this section we show data for both [eq] and [iid] clone selection methods, to illustrate that the performance of the [eq] method is better. All other sections use the [eq] method, unless stated otherwise.)

Results are shown in Figure 6. Panel (a) shows that on using sufficiently many clones and averaging over many computational realisations, the results are independent of Δt, as expected. Panel (b) shows how the variance of the estimate of $\mathcal{K}_L$ depends on Δt: the smaller is this variance, the less computational realisations are required to get accurate results, and hence the algorithm is more efficient. For the [eq] method, the variance is always smaller than for the [iid] method. Moreover, for [eq] method, the variance is monotonically increasing as a function of Δt. This implies that using a small value of Δt is desirable, from the point of view of accuracy. In practice, computational requirements mean that Δt should not be too small (Figure 6(c)), since the cloning stage of the algorithm incurs an overhead. For the [eq] method, this leads to a simple trade-off between accuracy (better for small Δt) and computational efficiency (better for large Δt). For the [iid] method, this trade-off is more complicated since accuracy may also be reduced by choosing smaller Δt.

To see why the [eq] method is more efficient, it is useful to consider the operation of the algorithm with $s = 0$. In this case, the most efficient algorithm is clearly to simulate each copy independently, so as to obtain $n_{\rm c}$ independent samples. Choosing equally-spaced α_j ensures that this does indeed happen (all the $\Upsilon^\beta(\Theta_k) = 1$ so one α_j lies in each interval). However, choosing the α_j independently means that some copies of the system are copied several times and some are deleted, because of the randomness inherent in the choice of α_j. The deletion of clones reduces the number of independent samples in the system and tends to increase the errors. The effect is the same for non-zero s and the problem also occurs in the continuous-time cloning algorithm as described in [29]. Hence, our conclusion is that choosing equally-spaced α_j as described in Sec. 4 is the more accurate of our two methods for selecting clones.

More generally, it is desirable – while always maintaining algorithmic accuracy – to minimise the number of clones that are deleted, since each deletion results in an (irreversible) loss of information, reducing the number of independent samples from $\tilde{P}$.

5. Dependence of the results on the parameters of the cloning algorithm

We have emphasised that the cloning algorithm gives accurate results only in the limit $n_{\rm c} \to \infty$. Characterising large deviations of the activity also requires that $t_{\rm obs} \to \infty$. Our ability to probe these limits depends on the available computational resource: it is therefore essential to characterise and understand the dependence of the algorithm's results on $n_{\rm c}$ and $t_{\rm obs}$, in order to assess whether the algorithm gives reliable results. For finite $n_{\rm c}$, a suitable choice of Δt is also essential to obtain accurate results. This section describes the dependence of the results on $n_{\rm c}$ and $t_{\rm obs}$.

We will find that the convergence of the algorithm depends significantly on the value of λ, particularly whether the system is in the homogeneous regime $\lambda < \lambda_c$ or the phase-separated regime $\lambda > \lambda_c$. The convergence also depends on the system size L, with large values of $n_{\rm c}, t_{\rm obs}$ being required when L is larger. When assessing the accuracy of our results, we focus on systematic errors, and we aim to achieve a relative error of less than 2% on our estimates of $\mathcal{K}(\lambda)$.

5.1. Convergence with respect to the time $t_{\rm obs}$: effects of long time scales

To analyse the dependence of the method on the choice of $t_{\rm obs}$, we consider the dependence of $\overline{k}(\lambda)$ on the observation time $t_{\rm obs}$. In this section, the number of clones used is sufficiently large that the results depend very weakly on $n_{\rm c}$: see Section 5.2. Results are shown in Figure 7. For $\lambda < \lambda_c$, Figure 7(a) shows that results depend weakly on $t_{\rm obs}$ as long as $t_{\rm obs} \gtrsim 10^3$. On the other hand, for $\lambda > \lambda_c$ there is a systematic dependence on $t_{\rm obs}$ even for $t_{\rm obs} > 10^3$. This message is confirmed by Figure 7(b) which shows that systematic errors from finite $t_{\rm obs}$ are larger for larger systems, but $t_{\rm obs} = 10^4$ seems to be large enough to achieve convergence even for $L = 80$. To verify this effect, we show (for $L = 80$) a fit to the asymptotic prediction [36, 37]

$$\bar{k}(\lambda)_{n_c,t_{\rm obs}} = k_\infty(1 + A/t_{\rm obs}) + O(t_{\rm obs}^{-2}) \tag{26}$$

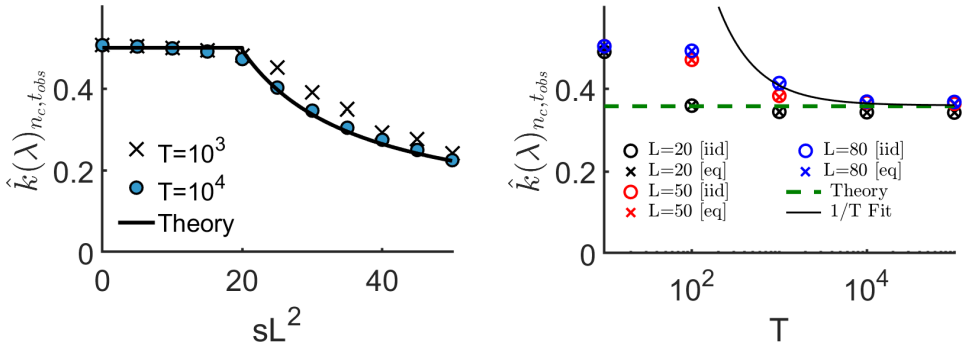

Figure 7: Estimates $\overline{k}(\lambda)$ of the activity $\mathcal{K}_L(\lambda)$ as a function of $t_{\rm obs}$ for various system sizes L. (a) $L = 50$ for various $\lambda, t_{\rm obs}$. (b) $\lambda = 28$ for various $t_{\rm obs}, L$. The "theory" line is $\mathcal{K}_* = \lim_{L\to\infty} \mathcal{K}_L(\lambda)$. We took $R = 10$, with $n_{\rm c} = 10^5$ for $L = 20, 50$ and $n_{\rm c} = 10^6$ for $L = 80$ (these $n_{\rm c}$ are large enough that results depend very weakly on $n_{\rm c}$). The fit uses equation (26) applied to the data for $L = 80$ [eq], using only the points at $t_{\rm obs} = 10^3, 10^4$. This fit captures the scaling for large $t_{\rm obs}$. See also Table 1.

	L=20		L=50		L=80	
	[iid]	[eq]	[iid]	[eq]	[id]	[eq]
Fit parameter A (exc. $t_{\rm obs} = 10^5$)	1.0	2.6	17.9	18.6	50.1	48.4
Fit parameter k_∞ (exc. $t_{\rm obs} = 10^5$)	0.3437	0.3434	0.3657	0.3620	0.3645	0.3605
Fit parameter k_∞ (incl. $t_{\rm obs} = 10^5$)	0.3433	0.3432	0.3649	0.3628	0.3666	0.3629
Difference in k_∞	0.1%	$< 0.1\%$	0.2%	0.2%	0.6%	0.7%

Table 1: Results of fitting the results in Fig. 7(b) to Equ (26), using data for $t_{\rm obs} \geq 10^3$. The first two rows are results of fitting just two points $t_{\rm obs} = 10^3, 10^4$. The third row shows the estimate of k_∞ when the final point at $t_{\rm obs} = 10^5$ is included in the fit. The asymptotic prediction for k_∞ is robust as more data are added, indicating that the data are consistent with the fit.

The fit (with parameters k_∞, A) is performed using the data points at $t_{\rm obs} = 10^3, 10^4$ and the resulting fit is consistent (up to our 2% tolerance) with the measured data point at $t_{\rm obs} = 10^5$. Thus, the results are consistent with the theory of [36, 37], and this fitting also provides accurate estimates of k_∞, provided one performs the fit using data points that are within the asymptotic regime. We show this fit for $L = 80$ and the [eq] clone selection method, the results for other cases are similar, as summarised in Table 1. Note however that data for small $t_{\rm obs}$ ($\lesssim 100$) are not at all consistent with the asymptotic prediction (26): one should remember that if data for large $t_{\rm obs}$ are not available, it may be difficult to assess which data are representative of the asymptotic regime and which should be excluded from the fit.

To understand the physical origin of the errors that arise from finite $t_{\rm obs}$, recall (16-18,25) and also the discussion of Section 2.4. The trajectories sampled by the algorithm have transient regimes at initial and final times, whose typical time scale is denoted by τ. In terms of (25), this means that for values of β that are outside the transient regimes, the variables K^β are all distributed according to the distribution $p_{\rm ave}$: in this case the average of K^β is independent of β and (assuming that $n_{\rm c}$ is large enough) this average is equal to $\Delta t L \mathcal{K}_L(\lambda)$. If the terms that are distant from the temporal boundaries dominate the sum in (25) then $\overline{k}(\lambda) = \mathcal{K}_L(\lambda)$, as required. However, when β is close to 1 or M, the average value of K^β depends on β and is not equal to $\Delta t L \mathcal{K}_L(\lambda)$. The inclusion of these terms in (25) means that $\overline{k}(\lambda) \neq \mathcal{K}_L(\lambda)$ in general: there are corrections of the order of $\tau/t_{\rm obs}$ coming from the transient regimes.

In fact, one can reduce these errors by excluding some of the transient terms from the sum. Figure 8 shows the transient regime that occurs for $(t_{\rm obs} - t) \lesssim \tau$, for two different observables. Two features are important here: the time scale associated with the transient regime, and the

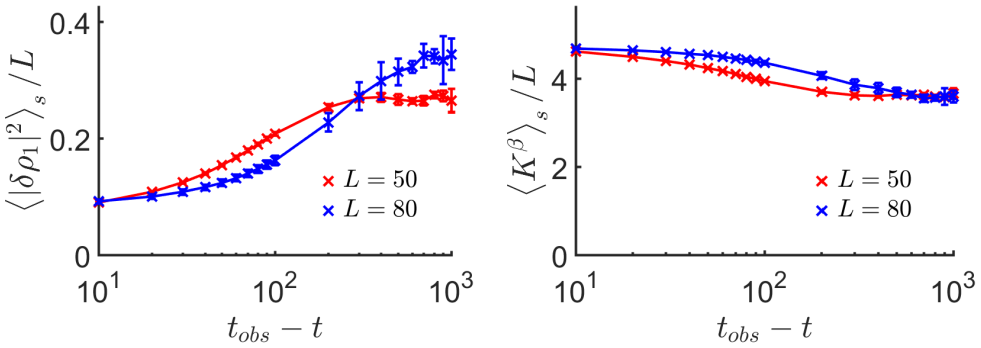

Figure 8: Time-dependent averages of the form $\langle F_t \rangle_s$ with F being either $|\delta\rho_1|^2$ (the squared modulus of the first Fourier component of the density) or K^β (which is the number of particle hops between times $t = t_\beta$ and $t_\beta + \Delta t$.) Parameters are $\lambda = 28$, $n_c = 10^6$, $t_{obs} = 10^4$, $\Delta t = 10$.

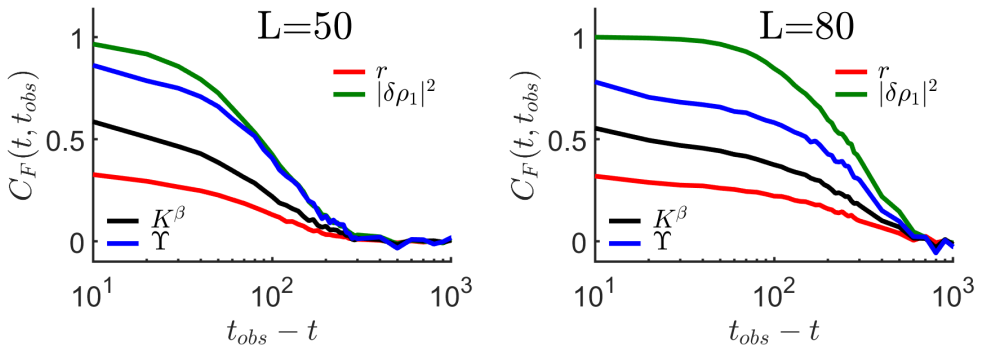

Figure 9: Autocorrelation function C_F, as a function of the general time t and the final time t_{obs}, for systems of $L = 50$ (left), 80 (right). The normalisations $c(K_\beta, t')$ are 946 ($L = 50$) and 1450 ($L = 80$). Data are shown for the observable F being the escape rate r, the first Fourier component of density $|\delta p_1|^2$, the activity per cloning interval K_β and the cloning factor Υ. Parameters are $t_{obs} = 10^4$, $n_c = 10^6$, $\lambda = 28$, $\Delta t = 10$.

difference between $\langle F_{t_{obs}} \rangle_s$ and F_∞ (which is the value when $t_{obs} - t \gg \tau$, recall (16)). For both observables, the order of magnitude of the transient time scale is $\tau \simeq 100$ for $L = 50$ and $\tau \simeq 300$ for $L = 80$. Obtaining an accurate estimate of $\mathcal{K}$ from (25) requires that $t_{obs} \gg \tau$, in order that the sum is dominated by terms that are outside the transient regime. Alternatively one can estimate $\mathcal{K}$ from the plateau value of $\langle K^\beta \rangle_s / L$ at large $t_{obs} - t$ (the value should be normalised by a factor of $\Delta t = 10$, for consistency with (25).) This amounts to excluding transient terms from the definition of $\hat{k}$, and does indeed give accurate results, at the expense of some post-processing. (We also note that the data in Fig. 8 are averaged over several independent runs of the algorithm, so identifying the plateau from the output of a single run of the algorithm may be non-trivial in practice.) In the following we continue to analyse the simplest estimator $\hat{k}$ but we note that excluding transient terms from the sum in (25) may well be a useful strategy for future applications of this algorithm.

Turning to the range of values of $\langle F_t \rangle_s$ as $t_{obs} - t$ is varied in Fig. 8, one sees that $|\rho_1|^2$ changes by more than a factor of 2, while the fractional changes in K^β are much smaller. This indicates that it is the long-wavelength density modes that respond most strongly to the field λ, consistent with Figure 3. This is as expected, because these slowly-relaxing long-wavelength modes are the origin of the dynamical phase transition [25].

To understand these dynamical fluctuations in more detail, we also consider the

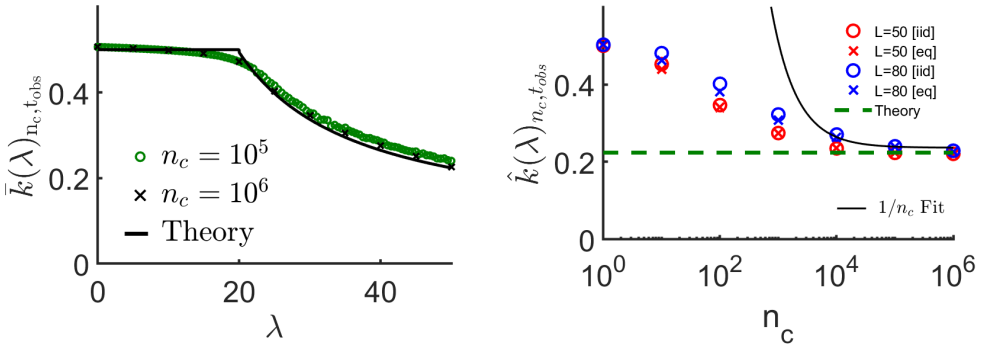

Figure 10: Estimates $\bar{k}_L(\lambda)$ of the activity, as the number of clones is increased. (a) Data for $L = 80$ and $n_c = 10^5, 10^6$, as λ is varied. The theory line is the result for $L \to \infty$. (b) Data for $\lambda = 50$ and $L = 50, 80$ as n_c is varied. In all cases, $t_{obs} = 10^4$ and $R = 10$. The fit uses equation (28) and is applied to the data for $L = 80$ [eq], using only the points at $n_c = 10^4, 10^5$. This fit captures the scaling for large n_c. See also Table 2.

autocorrelation function for observable F, defined as

$$C_F(t,t') = \frac{\langle F_{t'} F_t \rangle_s - \langle F_{t'} \rangle_s \langle F_t \rangle_s}{c_F(t')} \tag{27}$$

where $c_F(t') = \langle F_{t'} F_{t'} \rangle_s - \langle F_{t'} \rangle_s \langle F_{t'} \rangle_s$ is a normalisation factor that ensures that $C_F(t,t) = 1$. Since the transient time τ is comparable with the inverse of the spectral gap of the stochastic process [41], one expects $C_F(t,t')$ to be small if $t - t' \gg \tau$. Figure 9 shows results for several different observables, always with $t' = t_{obs}$ (this is the case for which good statistics are most easily obtained). For the Fourier component $F = |\delta\rho_1|^2$, one sees that $C_F(t, t_{obs})$ remains close to 1 until $t_{obs} - t \approx \tau$, after which point the density fluctuations at the two times decorrelate and the correlation decays. On the other hand, for other observables such as the activity K^β, the correlation $C(t, t_{obs})$ is significantly less than 1 already for $t_{obs} - t = 10$, indicating that the activity fluctuations have a "fast component" whose correlations decay quickly, as well as a slow component that is (presumably) correlated with the slow decay of large density fluctuations.

We also note that the activity correlations play a special role in the theory: one has $\chi(s) \sim L^{-1} c_F(t_{obs}) \int_0^{t_{obs}} C_{K^\beta}(t, t_{obs}) \mathrm{d}t$ [19], so χ can be large if either the prefactor c_F is large, or if the function C_{K^β} decays slowly to zero (so that the time integral becomes large). The data in Figure 9 (and its caption) show that the prefactor c_F scales roughly as L while the time τ is expected to scale as L^2 (the slowest time scale in the system is diffusive decay associated with wavelengths of order L). Hence $\chi \sim L^2$ which is consistent with $\mathcal{X} = O(1)$ and $\chi \sim L^2$ (these results are in the regime $\lambda > \lambda_c$, see also (23) and Figure 4).

The conclusions of this analysis are (i) that the longest relaxation time in the system controls the convergence with respect to t_{obs} and (ii) that these relaxation times can be revealed by explicitly computing transients and autocorrelation functions as in Figs. 8,9. We also emphasize that making these measurements are not only useful for verifying convergence of the algorithm: they also reveal the important physical effects at work in the biased trajectories of interest: for $\lambda > \lambda_c$, the dominant physical effect is that the density becomes inhomogeneous on the macroscopic scale, so that $\langle |\delta\rho_1|^2 \rangle_s$ diverges with system size. There is a slow time scale associated with this macroscopic inhomogeneity, which scales as $\tau \sim L^2$ and results in a large susceptibility χ. This long time scale necessitates a large t_{obs} in the cloning algorithm, since accurate estimates of observables like $\langle K \rangle_s$ require $t_{obs} \gg \tau$.

5.2. Population Size n_c Convergence

This section shows the convergence of the algorithm as n_c increases. In all cases we take $t_{obs} = 10^4$, which is large enough that results depend weakly on t_{obs}. Fig. 10 shows results, for a range of

	L=50		L=80	
	[iid]	[eq]	[iid]	[eq]
Fit parameter A (exc. $n_c = 10^6$)	130	140	340	270
Fit parameter k_∞ (exc. $n_c = 10^6$)	0.2221	0.2220	0.2369	0.2367
Fit parameter k_∞ (incl. $n_c = 10^6$)	0.2214	0.2204	0.2322	0.2311
Difference in k_∞	0.3%	0.7%	2.0%	2.4%

Table 2: Results of fitting the data in Fig. 10 to Equ (28). This is analogous to the analysis of Table 1 but the state point is different (so numerical values of k_∞ are different). The fits use data for $n_c \geq 10^4$. The first two rows are results of fitting just two points $n_c = 10^4, 10^5$. The third row shows the estimate of k_∞ when the final point at $n_c = 10^6$ is included in the fit. For $L = 80$, there is a small but systematic shift in the estimate of k_∞ as more data is included, indicating that subleading corrections to scaling are not negligible at $n_c = 10^4$.

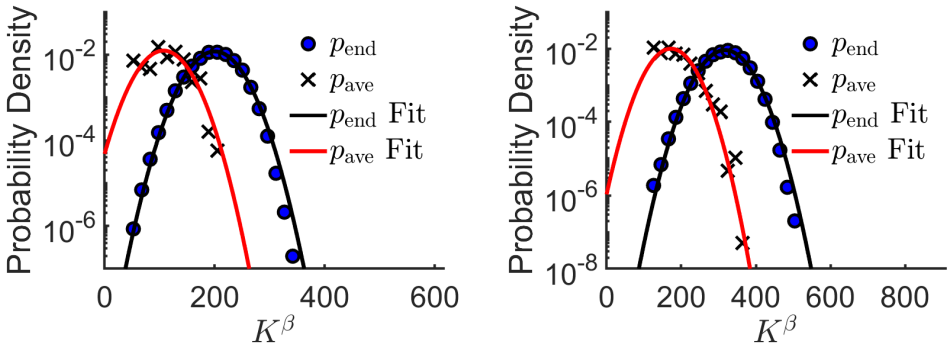

Figure 11: Distributions of the activity $K^\beta(\Theta_i)$ at $\beta = 970$ when $t_{\rm obs} = 10^4$ and $\Delta t = 10$. (a) $L = 50$; (b) $L = 80$. The data points are measured across $n_c = 10^6$ systems and Gaussian fits produced by measuring the average μ and variance σ^2 across the population. All results are at $\lambda = 50$.

system sizes and numbers of clones. From Fig. 10a, one sees that for the relatively large system size $L = 80$, a good estimate of $\mathcal{K}$ is available already for $n_c = 10^5$, but increasing to $n_c = 10^6$ clones and focussing on $s > s^*$, the results still depend on n_c, indicating that the large-n_c limit is not fully converged for $n_c = 10^5$. (For $s < s^*$, the results do not agree perfectly with theoretical prediction shown in Fig. 10a: this is because the theory applies only in the large-L limit, but these are finite systems. The important feature for convergence is not the agreement with the analytic theory, but whether the results depend significantly on n_c.) Fig. 10b shows results at the state point $\lambda = sL^2 = 50$, which is in the phase-separated regime (recall Fig. 2): one sees that the number of clones required to obtain accurate results is of order $10^5 - 10^6$, with the larger system requiring larger n_c.

To investigate this in more detail, Fig 10 shows a fitting analysis similar to that of Fig. 7, with a fit function

$$\bar{k}(\lambda)_{n_c,t_{\rm obs}} = k_\infty(1 + A/n_c) + O(n_c^{-2}), \tag{28}$$

analogous to (26). This is the large-n_c scaling form predicted by [36, 37]. As was the case for large $t_{\rm obs}$, a reasonably accurate extrapolation to the last data point ($n_c = 10^6$) can be obtained by a fit through the two previous data points ($n_c = 10^4, 10^5$). However, the quantitative analysis of Table 2 shows that for $L = 80$ there is a significant ($\sim 2\%$) shift in the estimated value of k_∞ when the data from $n_c = 10^6$ are included. This indicates that there are systematic deviations from the fit, presumably because data for $n_c = 10^4$ are not yet in the asymptotic regime described by (28).

To understand why these large numbers of clones are required, it is useful to return to the distributions $p_{\rm ave}$ and $p_{\rm end}$ defined in (17,18). We construct these distributions for the observable K^β, the number of particle hopping events in the time interval $[t_\beta, t_{\beta+1}]$. Results are shown in

Figure 11. Recall that the distribution $p_{\rm end}$ is representative of the transient regime ($t \approx t_{\rm obs}$), while the distribution $p_{\rm ave}$ is representative of the time-translation-invariant (TTI) regime. Accurate characterisation of the TTI regime is essential for obtaining accurate estimates of $\mathcal{K}_L$.

However, we note that the operation of the cloning algorithm means that the distribution $p_{\rm end}$ is sampled directly, but $p_{\rm ave}$ is obtained by a form of importance sampling from the distribution $p_{\rm end}$. In terms of Figure 11, this means that data is only available for $p_{\rm ave}$ over a restricted range, which is the range over which data is available for $p_{\rm end}$. If these two distributions have a low overlap, then the algorithm will not sample $p_{\rm ave}$ correctly: this is an important source of the systematic errors in Figure 10. In fact, this argument does not only apply to the distributions $p_{\rm end}$ and $p_{\rm ave}$ of the order parameter K^β – one requires that for any observable (say F) the distributions $p_{\rm ave}(F)$ and $p_{\rm end}(F)$ should overlap, otherwise the values of F sampled within the algorithm will be biased, leading (potentially) to systematic errors. The possible choices of observable include variants of K^β in which the averaging interval $[t_{\beta-1}, t_\beta]$ is replaced by some other interval, eg $[t_\beta - t_0, t_\beta]$: the overlap of $p_{\rm ave}$ and $p_{\rm end}$ for this distribution will likely have a non-trivial dependence on t_0. In this sense, significant overlap of the distributions shown in Fig. 11 is a necessary but not a sufficient condition for convergence: one requires in general that these distributions overlap for all observables F (otherwise the sampling of states within the TTI regime will be biased away from its correct distribution).

To estimate the significance of this effect (see also [46]), we focus for convenience on the distribution of K^β and suppose that both $p_{\rm ave}$ and $p_{\rm end}$ are described by (approximately) Gaussian distributions with mean values $\mu_{\rm ave}, \mu_{\rm end}$ and the same variance σ^2. (Note $\mu_{\rm ave} = \mathcal{K}_L$.) To be precise, define $g(x, \mu, \sigma) = \exp(-(x-\mu)^2/2\sigma^2)/\sqrt{2\pi\sigma^2}$ and suppose $p_{\rm end}(K) \approx g(K, \mu_{\rm end}, \sigma)$, and similarly for $p_{\rm ave}$. Also define a scaled version of the complementary error function as $G_\sigma(x) = \int_x^\infty g(y, 0, \sigma) {\rm d}y$. Given $n_{\rm c}$ clones drawn independently from $p_{\rm end}$, we expect to sample a range of K-values $(\mu_{\rm end} - A) < K < (\mu_{\rm end} + A)$ with $G_\sigma(A) = (1/n_{\rm c})$. We further assume that the measured $p_{\rm ave}$ is close to the true $p_{\rm ave}$ over the range over which data is available – this is certainly an approximation but it allows an estimate of the effect of the unsampled range on the results of the algorithm. With this approximation, the average value of K with respect to $p_{\rm ave}$ can be estimated as

$$\overline{k} \approx \frac{\int_{\mu_{\rm end}-A}^{\mu_{\rm end}+A} K g(K, \mu_{\rm ave}, \sigma) {\rm d}K}{\int_{\mu_{\rm end}-A}^{\mu_{\rm end}+A} g(K, \mu_{\rm ave}, \sigma) {\rm d}K} \tag{29}$$

Assuming as in Figure 11 that $\mu_{\rm end} > \mu_{\rm ave}$ (and σ is not too large), one may replace the upper limits in (29) by infinity. Writing $\Delta\mu = (\mu_{\rm end} - \mu_{\rm ave})$, this yields

$$(\overline{k} - \mu_{\rm ave}) \approx \sigma/\sqrt{2\pi} \cdot \frac{\exp(-(A-\Delta\mu)^2/2\sigma^2)}{G_\sigma(\Delta\mu - A)} \tag{30}$$

This error converges to zero as $A \to +\infty$, as it should do. However, for large $n_{\rm c}$ one has the scaling relation $A \sim \sigma\sqrt{\log n_{\rm c}}$ so A grows slowly with $n_{\rm c}$. The relevant dimensionless parameter for convergence is $X = (A - \Delta\mu)/\sigma$: this parameter is positive if the peak of the $p_{\rm ave}$ distribution is within the range of the sampled data: accurate results require $X \gtrsim 1$. This requires

$$n_{\rm c} \gtrsim \exp\left[(\Delta\mu/\sigma)^2\right]. \tag{31}$$

In general, one expects $\Delta\mu$ to increase as the biasing field increases; on the other hand σ is expected to depend weakly on the bias but (since K is extensive in space) one expects $\sigma \sim 1/L$ in large systems. Hence one expects that the number of clones required to achieve convergence increases exponentially in the system size and in the distance from equilibrium. For a similar argument, see [46, Section 5]. Based on the Gaussian fits in Figure 11 the lower bounds on $n_{\rm c}$ are 2.16×10^3 ($L = 50$) and 4.93×10^4 ($L = 80$): for high-accuracy estimates at these state points, we require $n_{\rm c} = 10^5$ and $n_{\rm c} = 10^6$ respectively, so high accuracy requires parameters significantly in excess of these lower bounds, consistent with this argument.

6. Computation

In this section, we discuss the performance of our computational implementation of the cloning algorithm. Given the large number of clones evolving independently, it is natural to use high-

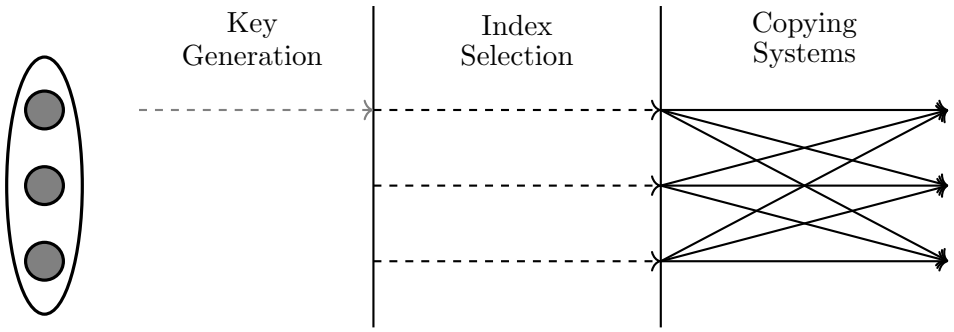

Figure 12: OpenMP implementation of the algorithm code run on one node (ellipse) and multiple threads (grey circles).

performance (parallel) computing methods to speed up the computations. However, the step where clones are copied and deleted involves a significant communication overhead, which can substantially reduce efficiency. Close to the phase transition ($\lambda = \lambda_c \approx 20$) the variance in activity is close to its maximum. This suggests that there should be a large variance in the $\Upsilon^{\beta}(\Theta_i)$, and a large amount of cloning activity. This makes the phase transition a useful place to test the algorithm.

We compare results obtained using MPI with simple serial computations and a shared-memory (OpenMP) implementation. We define the speed-up factor of an implementation m as

$$\mathcal{S}_m = \mathrm{RT}_s/\mathrm{RT}_m \tag{32}$$

where RT_m is the run time of implementation m and RT_s is the run time for a simple serial code. For a computation that uses n_t threads, we define the efficiency as

$$\mathcal{E}_m = \mathcal{S}_m/n_\mathrm{t} \tag{33}$$

Both the speed-up and the efficiency are depend in general on n_t. We describe an OpenMP implementation that is 99% efficient (but does not scale beyond a single computational node). For performance on multiple nodes, we use a method based on the message-passing interface (MPI) protocol, which achieves an efficiency of around 90% on computations distributed over 64 processors. We measure weak scaling performance up to 128 processors. All codes were written in C++ and will be available shortly after publication.

All results were obtained on (8-core) Intel Xeon E5-2650V2 Ivybridge processors running at 2.6 GHz. As a representative test case, we consider a system of size $L = 50$ at parameters at which the activity estimator $\hat{k}$ has converged, $t_\mathrm{obs} = 10^4$ and $n_\mathrm{c} = 10^5$. We average all results over 10 independent computations. This computation takes 5500 seconds to run on a single core. We present results for efficiency relative to this reference point. The shared memory implementation uses OpenMP with 16 threads on a dual-socket node. For the MPI implementation, we typically use four such nodes connected using Infiniband QDR with a total of 64 threads.

6.1. Serial and OpenMP Implementations

The simplest implementation is a serial code, which simulates the dynamics for each clone in turn. The cloning step involves copying and deletion of the different clones. This is achieved by each clone in the new population "pulling" its state (that is, the site occupancies, particle positions, etc) from a member of the old population ‡.

‡ The new and old population are stored separately in memory, so every cloning step involves n_c copy operations. If the new and old population are similar to each other, some of these copy operations could be avoided by updating the original population (in-place) instead of copying the new population from the old one. While this might improve efficiency in some cases, the benefits are negligible for the values of Δt considered here, since the new and old populations differ substantially, and the main computational effort comes from the system dynamics, not the copying. The advantage of copying the population instead of updating (in-place) is that it simplifies the implementation of the algorithm, since the copy operations are completely independent and happen in parallel.

Method	Speed-Up	Efficiency (%)
One clone per message	48.3	75.5
Many clones per message	55.2	86.2
Reduced communication	57.2	89.3

Table 3: Speed-ups and efficiencies relative to our serial implementation obtained by MPI implementations when using the equal spacing [eq] clone selection method with different amounts of packing when running on 64 processors distributed across 4 nodes. The results are obtained at $\lambda = 20$ and $L = 50$ with $t_{\rm obs} = 10^4, n_{\rm c} = 10^5$ and $\Delta t = 10$.

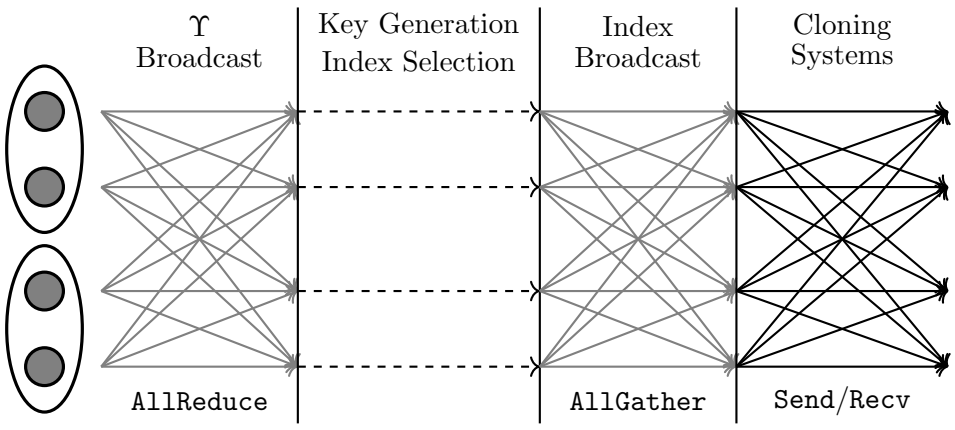

Figure 13: MPI implementation of the algorithm code run on two nodes (ellipses) and multiple threads (grey circles). Arrows are MPI communications, black arrows are pointwise and grey arrows are collective.

The choice of which member of the old population is copied into clone j of the new population is specified by the number α_j (see Sec. 4.1). One then has to find the value k that solves (24): this is achieved by a binary search. The *key* for this search is an ordered list of real numbers, in which the kth element is $\sum_{i=1}^{k} \Upsilon^{\beta}(\Theta_i)$. Random number generation uses the Mersenne Twister algorithm [47], with one instance of the random number generator for each clone. Except where otherwise stated (see below), this means that the results are fully reproducible, independent of whether the serial or parallel code is used.

In the OpenMP (shared memory) implementation, the clones are distributed equally over 16 threads on a single computational node. The dynamics are simulated as in the serial case. In the cloning step, the binary search key is constructed using a single thread, but the copying of clone states is done in parallel. A diagram summarising this process is shown in Figure 12. For our test case ($L = 50$, $t_{\rm obs} = 10^4$, $n_{\rm c} = 10^5$, $\lambda = 20$), the speed-up of the OpenMP implementation with respect to the serial code is 15.8, corresponding to an efficiency of 99%. Efficiencies at other state points are similar.

6.2. MPI Implementation

In order to exploit higher levels of parallelisation, we also use a distributed memory (MPI) implementation, which is illustrated in Figure 13. The direct generalisation of the OpenMP code gives rise to an implementation that we call "simple communication". In this case, the MPI method produces identical output to the OpenMP method. We also describe a method with reduced communication, in which case the results are statistically equivalent (but not identical), because the systems being simulated are distributed differently among the relevant threads, and hence the instance of the random number generator that is used to simulate a particular system is different.

6.2.1. Simple communication. In the dynamical stage of the algorithm, the clones are distributed evenly across the threads. In the cloning step, each thread sends the $\Upsilon^{\beta}(\Theta_i)$ for its clones to all other threads (this is an MPI `AllReduce` communication). Each thread constructs the binary key independently (this computation is therefore duplicated for each thread, but this is more efficient than calculating it on one thread and broadcasting it to all others). With the binary key in place, each clone in the new population uses (24) to decide from which clone to pull its state (the value of d in (24) is produced by thread 0 and broadcast to all threads after the key generation). Next, the indices of the clones that are required by each thread are communicated to all other threads (MPI `AllGather` operation); based on that information, the clone states are sent between the nodes, as required (using many MPI `Send` operations). In the simplest case, the state of each clone is sent using a single MPI message (we call this "one-clone-per-message"). Alternatively, all information to be exchanged between each pair of threads can be packed into a single message ("many-clones-per-message").

From Table 3, these two methods achieve efficiencies of 75-86% on systems of $n_{\mathrm{t}} = 64$ threads, with the many-clones-per-message method being more efficient. We emphasise that the amount of information being exchanged between each thread is identical in this case, so the increased efficiency comes from packing the same information into a smaller number of messages. For one-clone-per-message, the total number of messages is $n_{\mathrm{c}} = 10^5$; for many-clones-per-message, each thread receives at most one other message from each other thread, so the total number of messages is at most $n_{\mathrm{t}}(n_{\mathrm{t}} - 1) = 4032$ (see Sec. 6.2.3, below). This is a reduction by more than an order of magnitude in the number of messages, which significantly improves performance.

6.2.2. Reduced communication. The simple communication method is somewhat inefficient because in some situations, a particular thread (A) may send several clones to some other thread (B), but thread B also sends several clones back to thread A. To avoid this redundancy, we use a cancellation procedure where each thread preferentially pulls states from clones that are on the same thread. (This is achieved by pairwise cancellation of clones that have been designated by (24) to be sent between threads; the resulting new population is identical but its partitioning among the threads is different.) With this "reduced communication" method, clones are sent either from A to B or from B to A (instead of sending one message in each direction). This typically reduces the number of messages by almost a factor of 2, and the size of the resulting messages by a factor of 9. Table 3 shows that this leads to an improvement in efficiency.

6.2.3. Communication Patterns To illustrate the communications pattern, Figure 14 shows the number of messages sent and received by each thread in a typical cloning step at $t = t_{\mathrm{obs}}/2 = 5000$. In the simple implementation, each thread typically sends around 30 clones to each other thread and receives a similar number. In the reduced communication implementation, there are less than half as many MPI communications and those that occur contain about 5 systems.

6.2.4. Weak Scaling For a fixed number of threads n_{t}, the run time for each algorithm scales linearly with the number of clones n_{c}. A *weak scaling* analysis is one in which the problem size (in this case n_{c}) is increased, with a proportionate increase in the computational resource (the number of nodes $n_N = n_{\mathrm{t}}/16$). In the best (most efficient) case, this leads to a run-time that remains constant as the problem-size n_{c} increases. In general, one expects the run-time to increase with problem size, with a smaller increase corresponding to a more efficient algorithm.

In Figure 15 we measure the weak-scaling of a system at $L = 50$ with enough clones ($nc/n_N = 10^5$) that the algorithm is converged when the code is run on one node. The value of the bias ($\lambda = 20$) that we consider is very close to the phase transition and the value of the estimator $\hat{k}$ depends only very weakly on n_{c}. We show results for both the [eq] and [iid] clone selection methods, but the results are similar in both cases.

As the problem size increases, one observes a gradual increase in run time: for the most efficient algorithm, this increases by approximately 5% while the number of clones has increased by a factor of 8. This shows that parallel-computing platforms significantly reduce the wall-time required to perform large computations, although (as expected) the method is most efficient when run on just a few nodes, since the communication overhead is lower in that case. The gradual

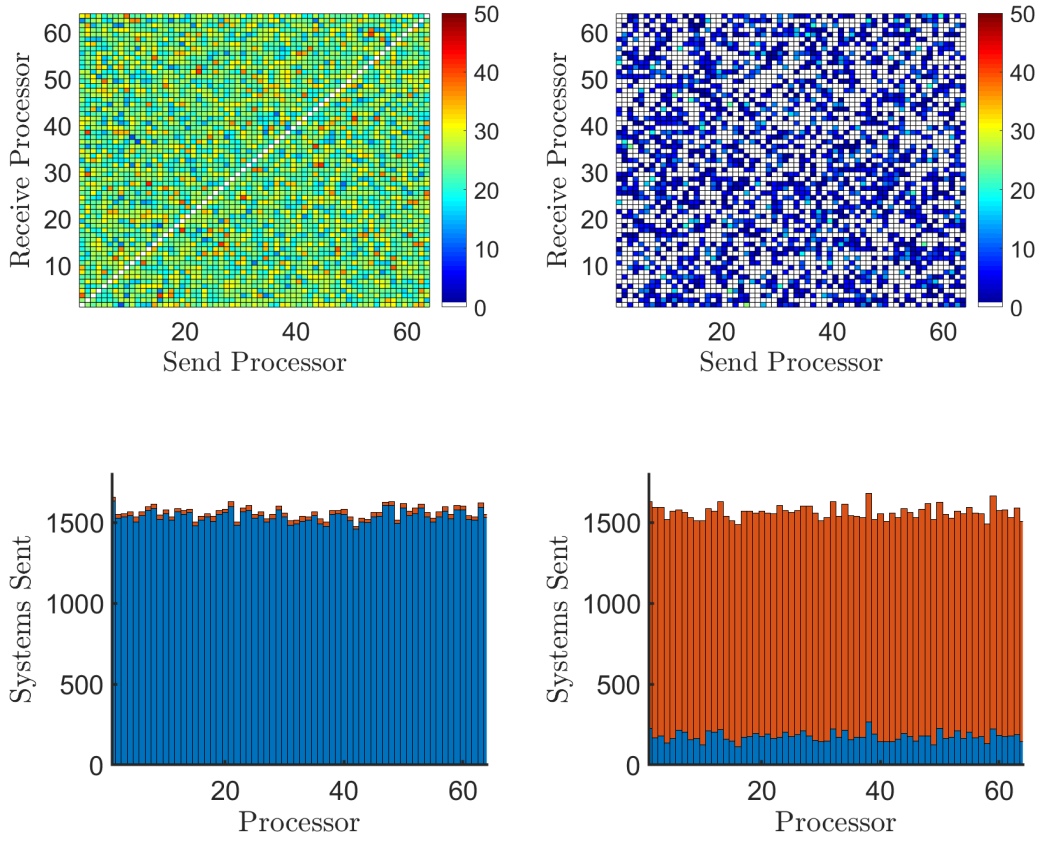

Figure 14: The number of systems sent between each pair of processors using MPI when evolving $n_c = 10^5, t_{obs} = 10^4$ and $L = 50$. Simple communications (*left*) and reduced communications (*right*) using the independent [iid] clone selection method. These communications are in the 500^{th} cloning interval ($t = 5000$) and the cloning intervals are $dt = 10$ units of time. The algorithm runs at a bias $\lambda = 20$. *Bottom Row:* The number of systems sent by each processor using MPI (blue) and those whose states are simply copied (orange).

increase in run-time with n_N does not indicate any optimum value for n_N in this case nor any cutoff beyond which the method becomes inefficient: the code is performing well across this range of n_N (which corresponds to at most 128 parallel threads).

7. Conclusion

We have analysed the finite-size scaling of the dynamical phase transition in the SSEP. Our results show that the cloning algorithm can demonstrate convergence of the limit of infinite system-size $L \to \infty$ in this problem, although of order 10^6 clones are required to achieve this. By analysis of the p_{ave} and p_{end} distributions of the activity we can understand the number of clones required for convergence. We also analysed the time t_{obs} required for convergence of the limit $t_{obs} \to \infty$, by measuring of the transient decay of several observables, and their associated autocorrelation functions.

We have achieved computational efficiencies of 99% using an OpenMP implementation of the algorithm, although this requires shared memory for all threads and so can be used on a single node. We also used an MPI implementation, so that the algorithm can be scaled across multiple nodes. Optimising MPI communication to use as few messages as possible significantly increases the algorithmic performance. We have achieved this by point-wise reduction of the MPI

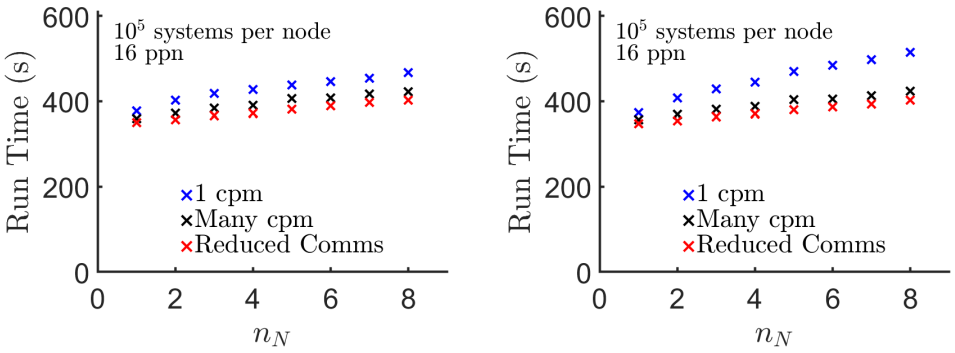

Figure 15: The weak scaling of the run time of the code for both clone selection methods [iid] (left) and [eq] (right). We fix $n_c/n_N = 10^5$ and the code is run at $t_{\text{obs}} = 10^4$, $L = 50$, $\lambda = 20$ and $dt = 10$. We vary the number of clones per message (cpm) and reduce the communications.

communications between pairs of processors and packing multiple systems into each MPI message. By measuring the weak-scaling of run time across multiple processors we find that there is a slow drop off in efficiency.

The implementation of the cloning stage of the algorithm can affect the errors in the method. We have made a comparison of two methods for selecting which systems are cloned, one of these methods reduces the number of systems that are deleted. Whilst both methods produce similar systematic errors and require similar computation time, we have found that the method which reduces the number of system deletions incurs much smaller statistical errors.

As the cloning algorithm becomes more widely used [35, 38, 42], we hope that these methods for error analysis and parallel computing implementation will be useful for the growing effort into research on large deviations of time-averaged quantities.

Acknowledgments

RLJ thanks Takahiro Nemoto and Vivien Lecomte for useful discussions about the population dynamics algorithm. We also thank the anonymous referees for insightful comments and suggestions. Computations were performed on the HPC service at the University of Bath.

8. References

[1] Touchette H 2009 *Phys. Rep.* **478** 1–69
[2] Gretener P 1967 *AAPG Bulletin* **51** 2197–2206
[3] Easterling D R, Meehl G A, Parmesan C, Changnon S A, Karl T R and Mearns L O 2000 *Science* **289** 2068–2074
[4] Derrida B 2007 *J. Stat. Mech.* **2007** P07023
[5] Garrahan J P, Jack R L, Lecomte V, Pitard E, van Duijvendijk K and van Wijland F 2007 *Phys. Rev. Lett.* **98** 195702
[6] Hedges L O, Jack R L, Garrahan J P and Chandler D 2009 *Science* **323** 1309–1313
[7] Speck T, Malins A and Royall C P 2012 *Phys. Rev. Lett.* **109**(19) 195703
[8] Weber J K, Jack R L and Pande V S 2013 *Journal of the American Chemical Society* **135** 5501–5504
[9] Pinchaipat R, Campo M, Turci F, Hallett J E, Speck T and Royall C P 2017 *Phys. Rev. Lett.* **119**(2) 028004
[10] Bolhuis P G, Chandler D, Dellago C and Geissler P L 2002 *Ann. Rev. Phys. Chem.* **53** 291–318
[11] Allen R J, Warren P B and ten Wolde P R 2005 *Phys. Rev. Lett.* **94** 018104
[12] Giardina C, Kurchan J and Peliti L 2006 *Phys. Rev. Lett.* **96** 120603
[13] Huber G A and Kim S 1996 *Biophysical Journal* **70** 97–110
[14] Zhang B W, Jasnow D and Zuckerman D M 2010 *J. Chem. Phys.* **132** 054107
[15] Bouchet F and Reygner J 2016 *Annales Henri Poincaré* **17** 3499–3532
[16] Johnson T H, Elliott T, Clark S R and Jaksch D 2015 *Physical review letters* **114** 090602
[17] Johnson T H, Clark S R and Jaksch D 2010 *Phys. Rev. E* **82**(3) 036702
[18] Jack R L, Garrahan J P and Chandler D 2006 *The Journal of chemical physics* **125** 184509
[19] Garrahan J P, Jack R L, Lecomte V, Pitard E, van Duijvendijk K and van Wijland F 2009 *Journal of Physics A: Mathematical and Theoretical* **42** 075007

[20] Weber J K, Jack R L, Schwantes C R and Pande V S 2014 *Biophysical journal* **107** 974–982
[21] Mey A S, Geissler P L and Garrahan J P 2014 *Physical Review E* **89** 032109
[22] Tailleur J and Kurchan J 2007 *Nature Physics* **3** 203–207
[23] Appert-Rolland C, Derrida B, Lecomte V and van Wijland F 2008 *Phys. Rev. E* **78** 021122
[24] Lecomte V, Garrahan J P *et al.* 2012 *Journal of Physics A: Mathematical and Theoretical* **45** 175001
[25] Thompson I R and Jack R L 2015 *Physical Review E* **92** 052115
[26] Bodineau T and Derrida B 2005 *Physical Review E* **72** 066110
[27] Bodineau T and Derrida B 2007 *Comptes Rendus Physique* **8** 540–555
[28] Bodineau T, Lecomte V and Toninelli C 2012 *Journal of Statistical Physics* **147** 1–17
[29] Lecomte V and Tailleur J 2007 *Journal of Statistical Mechanics: Theory and Experiment* **2007** P03004
[30] Grassberger P and Procaccia I 1983 *Phys. Rev. A* **28**(4) 2591–2593
[31] Grassberger P 2002 *Computer Physics Communications* **147** 64–70
[32] Anderson J B 1975 *J. Chem. Phys.* **63** 1499–1503
[33] Pitard E, Lecomte V and van Wijland F 2011 *EPL* **96** 56002
[34] Hurtado P I, Espigares C P, del Pozo J J and Garrido P L 2014 *J. Stat. Phys.* **154** 214–264
[35] Nemoto T, Jack R L and Lecomte V 2017 *Phys. Rev. Lett.* **118**(11) 115702
[36] Nemoto T, Guevara Hidalgo E and Lecomte V 2017 *Phys. Rev. E* **95**(1) 012102
[37] Guevara Hidalgo E, Nemoto T and Lecomte V 2017 *Phys. Rev. E* **95**(6) 062134
[38] Ray U, Kin-Lic Chan G and T Limmer D 2017 arXiv:1708.00459
[39] Nemoto T, Bouchet F, Jack R L and Lecomte V 2016 *Physical Review E* **93** 062123
[40] Chetrite R and Touchette H 2015 *Ann. Henri Poincare* **16** 2005–2057
[41] Jack R L and Sollich P 2010 *Progress of Theoretical Physics Supplement* **184** 304–317
[42] Carollo F, Garrahan J P, Lesanovsky I and Perez-Espigares C 2017 Phys. Rev. E **96**, 052118
[43] Baek Y, Kafri Y and Lecomte V 2017 *Phys. Rev. Lett.* **118**(3) 030604
[44] Bortz A B, Kalos M H and Lebowitz J L 1975 *J. Comp. Phys.* **17** 10–18
[45] Tailleur J and Lecomte V 2009, AIP Conference Proceedings **1091**, 212
[46] Hurtado P I and Garrido P L 2009 *J. Stat. Mech.* **2009** P02032
[47] Matsumoto M and Nishimura T 1998 *ACM Trans. Model. Comput. Simul.* **8** 3–30